Department Informatik
Lehrstuhl für Informatik 3 (Rechnerarchitektur)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
24.09.2015
Klausur zu

Grundlagen der Rechnerarchitektur und -organisation

.....	.....
Matrikelnummer	Geburtsdatum
.....	.....
Name	Vorname

- Es sind *keine* Hilfsmittel erlaubt!
- Legen Sie den Ausweis (mit Lichtbild!) griffbereit auf den Platz!
- Dieses Aufgabenheft umfasst 18 Seiten. Überprüfen Sie die Vollständigkeit!
- Gesondert beigelegte Blätter werden nicht bewertet.
- Schreiben Sie deutlich! Unleserliches wird nicht bewertet!
- Es darf nicht mit der Farbe rot geschrieben werden!
- Offensichtlich falsche oder überflüssige Antworten können zu Punktabzug führen!
- Begründen Sie Ihre Antworten!

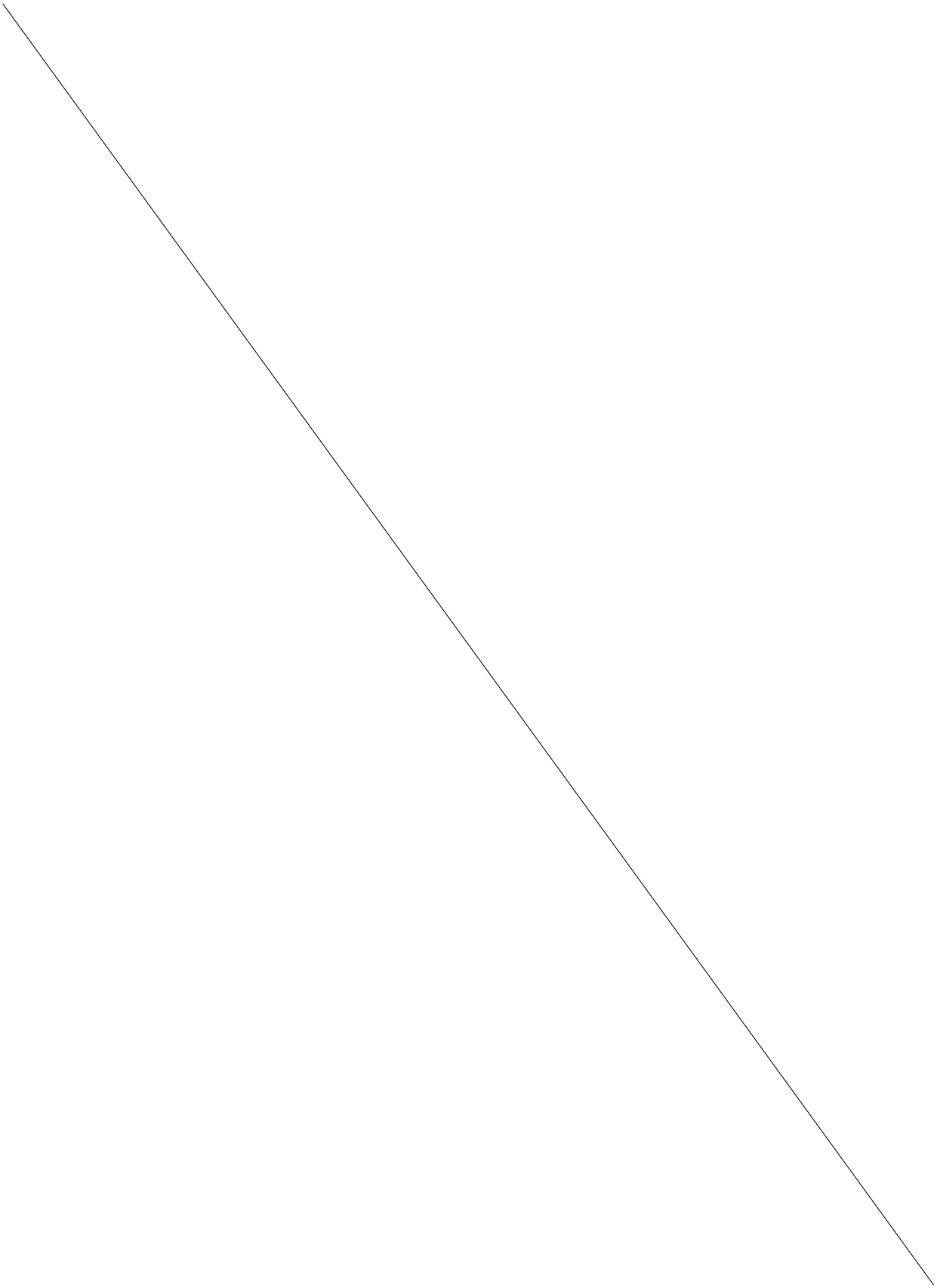
Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen
- die Kenntnisnahme der obigen Informationen.

Erlangen, den 24.09.2015 .....
(Unterschrift)

Aufgabe	1	2	3	4	5	6	7	8
max. Punktzahl	9	16	12	8	12	13	9	11
erreichte Punktzahl								

Summe	/90		
Bonus	/15		
Gesamt		Note	



Aufgabe 1: Allgemein — 9 Punkte

Kreuzen Sie an. Punktabzug bei falschen Antworten!

1. Beziehen sich die folgenden Aussagen auf die Programmierersicht (Ungerer: Befehlsarchitektur) eines Prozessors? 3 Punkte

ja	nein	
☐	☐	Es stehen 16 Universalregister für arithmetische Operationen zur Verfügung.
☐	☐	Die Universalregister sind 64 Bit breit.
☐	☐	Der Prozessor ist in 22 nm-Technologie gefertigt.
☐	☐	Es gibt Instruktionen zur Fließkommaaddition und –multiplikation.
☐	☐	Fließkommabefehle werden durch Integerarithmetik emuliert.
☐	☐	Der Prozessor verwendet eine superskalare Pipeline.

Welche der folgenden Aussagen sind wahr, welche falsch?

2. Ein-/Ausgabe 3 Punkte

wahr	falsch	
☐	☐	Der Datencache des Prozessors soll Zugriffe auf IO-Geräte beschleunigen.
☐	☐	PIO hat zur Datenübertragung zur CPU eine geringere Latenz als DMA.
☐	☐	Lesen aus Memory-Mapped-I/O-Geräten erfordert andere Maschineninstruktionen als Lesen aus dem RAM.
☐	☐	DMA erlaubt der CPU direkten Speicherzugriff.
☐	☐	Bei PCI-Express wird ein hoch getakteter, paralleler Bus verwendet.
☐	☐	Interrupt-Handler sind oft Teil eines Gerätetreibers.

3. Speicher 3 Punkte

wahr	falsch	
☐	☐	Bei DDR-SDRAM ist der Speicher(chip-)takt immer mindestens doppelt so hoch wie bei SDRAM.
☐	☐	Bei Little-Endian steht das niederwertigste Byte an der niedrigsten Speicheradresse.
☐	☐	Die Byte-Reihenfolge (Endianess) ist nicht bei allen Datentypen (char, int, . . .) relevant.
☐	☐	Speicherschutz ist nicht effizient ohne CPU-Unterstützung möglich.
☐	☐	Bei einem vollassoziativen Cache ist die Ersetzungsstrategie entscheidend für die Trefferwahrscheinlichkeit.
☐	☐	Ein 8-fach assoziativer Cache verwendet einen 3 Bit breiten Index.

Aufgabe 2: Assembler — 16 Punkte

Gegeben sei das folgende AT&T-Assemblerprogramm für einen 32-Bit Intel-x86-Prozessor:

```
1  .text
2  foo:
3          movl    4(%esp), %eax
4          pushl   %eax
5          call    baz
6          addl    $4, %esp
7          call    bar
8          ret
9  bar:
10         pushl   $2
11         call    baz
12         addl    $4, %esp
13         incl    %eax
14         ret
15 baz:
16         movl    $1, %eax
17         addl    4(%esp), %eax
18         ret
19 .globl main
20 main:
21         call    bar
22         movl    %eax, %ecx
23         pushl   $5
24         call    foo
25         addl    $4, %esp
26         addl    %ecx, %eax
27         ret
```

1. Für welche Klasse von Befehlssatzarchitekturen wurde das Programm wahrscheinlich geschrieben? Begründen Sie Ihre Entscheidung anhand mindestens einer konkreten Instruktion!
   1 Punkt

2. Was versteht man unter „caller-save“ und „callee-save“ in Bezug auf Register? 2 Punkte

3. Wieviel Byte auf dem Stack benötigt das Programm zur Laufzeit maximal?

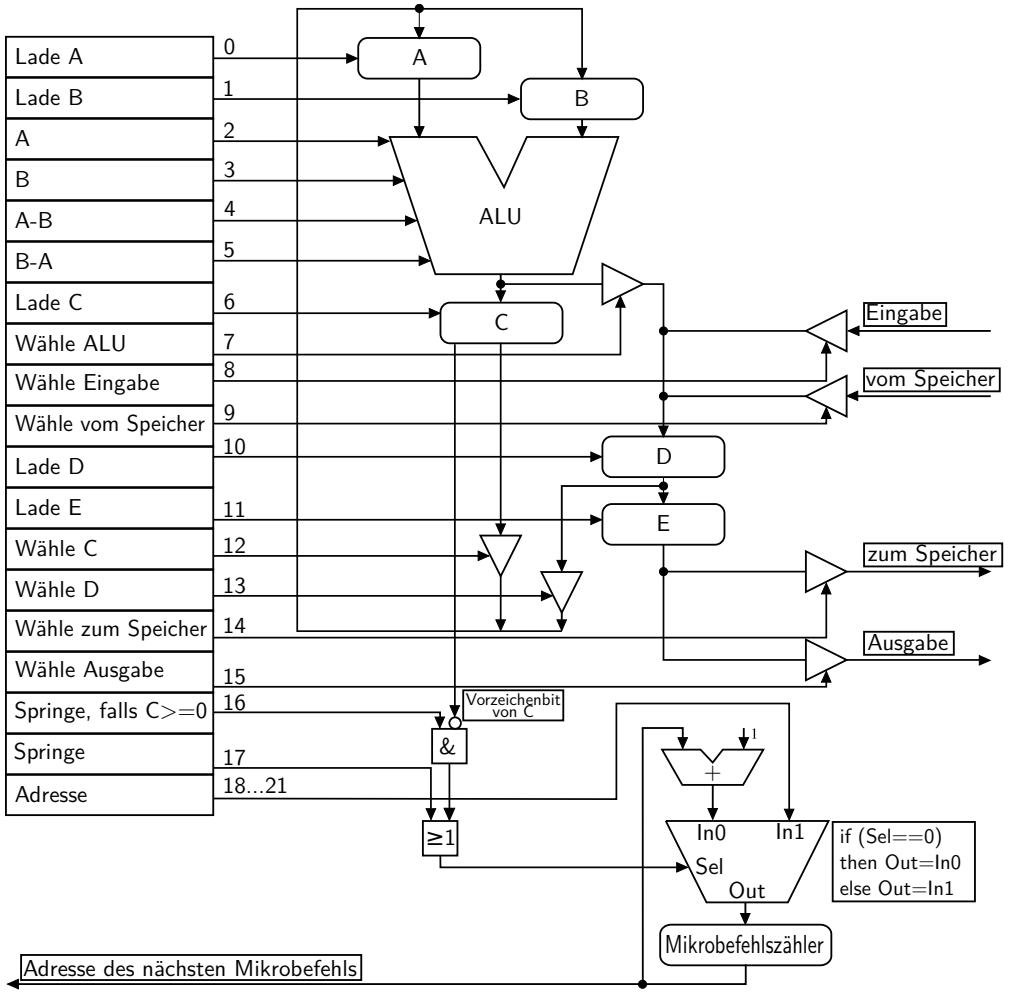
   Die Rücksprungadresse der Funktion `main` befindet sich zu Beginn auf dem Stack und soll mitgezählt werden. *(Begründung!)* 6 Punkte

4. Wie könnte der ursprüngliche C-Code ausgesehen haben? Achten Sie auf Parameter und Rückgabewerte. Alle Variablen sind vom Typ `int`. 6 Punkte

5. Welchen Zahlenwert gibt die Funktion `main` zurück? 1 Punkt

Aufgabe 3: Mikroprogrammierung — 12 Punkte

Gegeben sei der folgende Teil einer CPU:



1. Was versteht man unter vertikaler Mikroprogrammierung und welchen Vorteil hat sie im Vergleich zu horizontaler? *(Stichpunkte!)* 2 Punkte

2. Schreiben Sie ein Mikroprogramm für obige CPU, das nacheinander zwei ganze Zahlen von der Eingabe liest und den Betrag der Differenz *abs*(Zahl1 − Zahl2) in die Ausgabe schreibt.

   Vermeiden Sie unnötige Befehle! Alle Register sind zu Beginn „0“ und dürfen überschrieben werden.

   Verwenden Sie die folgende Tabelle und tragen Sie jeweils auch die Bedeutung (z.B. D → E) in die Spalte „Erklärung“ ein. Leer gelassene Steuerleitungen entsprechen dem Wert „0“, bei Sprüngen muss die Zieladresse explizit angegeben werden! Das niederwertigste Bit des Sprungziels entspricht Steuerleitung 21.

   Hinweis:
   Überlegen Sie sich zunächst den Ablauf in Pseudocode (wird nicht bewertet)! 6 Punkte

(Die Tabellenlänge entspricht nicht der erwarteten Mikroprogrammlänge!)

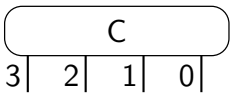
Adr.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Erklärung
0																							
1																							
2																							
3																							
4																							
5																							
6																							
7																							
8																							
9																							
10																							
11																							
12																							
13																							
14																							
15																							

3. Skizzieren Sie, wie die folgenden Sprungbedingungen realisiert werden könnten.

   Das Register `C` enthält eine 4 Bit breite Zahl im Zweierkomplement. Es soll eine logische „1“ am Ausgang der Schaltung generiert werden, wenn die jeweilige Bedingung erfüllt ist.

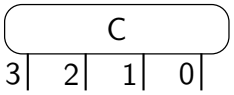
   Verwenden Sie einfache Logikgatter (AND &, OR ≥1, NOT 1○). 4 Punkte

a) $C < 0$



Ausgang

b) $C = 0$



Ausgang

Aufgabe 4: Umformung — 8 Punkte

Gegeben sei folgender Hochsprachencode:

```
1 while ( ( !(c == EOF) && !(c == 'q' && skip == 0) ) && quit == 0 ) {
2   c = getchar();
3   if (skip_magic == 1) continue;
4   do_magic(c);
5 }
```

Formen Sie das C-Programm so in if-goto-Darstellung um, dass es sich möglichst leicht in Assembler übersetzen lässt.

Verändern Sie dabei weder die Auswertungsreihenfolge noch die eigentlichen Operationen. Achten Sie auf die Kurzschluss-Semantik!

Aufgabe 5: Parallelverarbeitung — 12 Punkte

Gegeben sei das folgende Programmfragment in AT&T-Syntax:

```
1 | movl $1, %eax
2 | movl $2, %ebx
3 | movl $3, %ecx
4 | addl %eax, %edx
5 | addl %ebx, %edx
```

1. Ermitteln Sie nachvollziehbar den tatsächlichen Speedup einer Architektur mit einer einfachen Pipeline, ohne Sprungvorhersage und ohne Forwarding, mit den Stufen

   - Befehl holen und decodieren (HD)
   - Operanden holen (OH)
   - Ausführen (EX)
   - Rückschreiben (WB)

   bei Ausführung der obigen Instruktionen, im Vergleich zu einer streng sequentiellen Befehlsausführung ohne Pipelining.

   Befehle mit Datenabhängigkeiten warten in der OH-Phase. Die Pipeline ist zu Beginn leer. Das Programmfragment ist fertig, wenn der letzte Befehl die Pipeline verlässt. Die Ausführungsdauer der einzelnen Stufen ist identisch. 5 Punkte

2. Wie hoch ist der Speedup, wenn die Architektur um Forwarding erweitert wird?
   (Begründung!) 2 Punkte

3. Unterscheiden sich Maschinenprogramme für VLIW-basierte und superskalare Architekturen?
   (Begründung!) 2 Punkte

4. Nennen Sie drei verschiedenen Arten von Multithreading auf einer CPU und geben Sie an, wieviele Threadkontexte (Befehlszähler, Universalregistersätze, Flags-Register) jeweils gleichzeitig aktiv sind.

   Begünden Sie jeweils kurz Ihre Antwort. 3 Punkte

Aufgabe 6: Paging — 13 Punkte

Eine CPU biete eine Memory-Management-Unit mit folgenden Eigenschaften:

- zweistufige Adresstabellen
- je 1024 Einträge zu je 4 Bytes in den Tabellen
  - Bit 31-12: höherwertige Bits der physikalischen Adresse
  - Bit 11-4: unbenutzt
  - Bit 3: Cache-disable-Bit
  - Bit 2: Execute-enable-Bit
  - Bit 1: Write-enable-Bit
  - Bit 0: Present-Bit
- Pages zu je 4 KiB Größe

Folgendes soll für ein Programm gelten:

- im virtuellen Adressbereich `0x00 03 00 00` bis `0x00 03 ff ff` kann auf nicht ausführbare Daten zugegriffen werden,
- im virtuellen Adressbereich `0x00 20 00 00` bis `0x00 2f ff ff` ist der Speicher der Grafikkarte eingeblendet,
- im virtuellen Adressbereich `0x00 e0 00 00` bis `0x00 e0 0f ff` kann auf schreibgeschützten Programmcode zugegriffen werden,
- alle anderen Bereiche sollen Zugriffsfehler auslösen.

Freier Arbeitsspeicher ist ab der physikalischen Adresse `0x00 c0 00 00` verfügbar, die Grafikkarte ist ab Adresse `0x00 01 e0 00` eingeblendet.

Hinweis:
Ergebnisse dürfen hexadezimal angegeben werden.

1. Zeichnen Sie eine Tabellenhierarchie, die obige Anforderungen erfüllt! Leere Tabelleneinträge sind nicht „Present“ und müssen nicht extra gekennzeichnet werden. Lassen Sie keine Bereiche im physikalischen Adressraum unnötig leer.

   (Bearbeitung auf der nächsten Seite!) — 10 Punkte

Fortsetzung von Aufgabe 5.1

2. Warum verwendet man beim Paging bei gleicher Speichergröße normalerweise nicht ausschließlich

   - sehr kleine Seiten (<<1 KiB)?

   - sehr große Seiten (>>1 MiB)?

2 Punkte

3. Warum wird das Present-Bit nicht im TLB gespeichert? 1 Punkt

Aufgabe 7: Peripherie — 9 Punkte

1. Was ist charakteristisch für Interrupts und wofür werden sie benötigt? *(Stichpunkte!)*
   2 Punkte

2. Welche drei Formen der Datenübertragung zwischen Ein-/Ausgabegeräten und der CPU oder dem Arbeitsspeicher wurden behandelt?

   Geben Sie jeweils ein Beispiel für eine Datenübertragung, und erläutern Sie, worin der Vorteil der entsprechenden Variante für diese Übertragung liegt. 6 Punkte

3. Was versteht man unter dem Burst-Modus bei Datenübertragungen? Was ist der Vorteil?
   1 Punkt

Aufgabe 8: Cache — 11 Punkte

1. Welche Rolle spielt die Ersetzungsstrategie bei direktabbildenden Caches im Vergleich zu vollassoziativen? 2 Punkte

2. Eine CPU besitze einen 32 KiB großen direktabbildenden Cache mit einer Blockgröße von 64 Byte. Der Cache sei zu Beginn leer. Es werden insgesamt mehr Daten adressiert, als in den Cache passen, pro Cache-Block erfolgt nur ein einziger Zugriff.

   - Beim wievielten Speicherzugriff findet frühestens eine Verdrängung aus dem Cache statt?

   - Beim wievielten findet spätestens eine Verdrängung statt?

   - Wie nennen sich die Misses, die bei obigen Zugriffen dabei auftreten?

   3 Punkte

3. Eine 32-Bit CPU verwende einen 2-fach assoziativen Cache mit insgesamt 8 Blöcken. Jeder Block umfasse 16 Byte. Der Cache sei zu Beginn leer. Es wird jeweils der Eintrag verdrängt, der am längsten nicht verwendet wurde (LRU).

   Ein Programm liest nacheinander *jeweils 4 Byte* von den folgenden Adressen:
   `0xc, 0x20, 0x60, 0x1a8, 0x1bc`.

   Der relevante Speicherbereich hat folgenden Inhalt:

```
0x00 00 00 00: 34 2f 6d a4 56 96 9b ec  a5 f0 73 51 a2 4b 39 5a
0x00 00 00 10: 95 14 0f e0 66 48 de 8e  4a b7 cb 38 77 f2 44 f8
0x00 00 00 20: 00 a4 d5 31 38 32 81 f4  b1 c3 de 33 e0 9f fb ff
0x00 00 00 30: 2e 54 60 9d aa 85 ba ad  28 ad 92 d7 b7 10 1f 21
...
0x00 00 00 60: 48 e6 87 60 a5 bf 20 f6  e6 24 cd 4b 40 79 c7 c8
0x00 00 00 70: ee dd d7 23 2d ab ab e4  6c 8c bd 8f cc 10 1f 7b
...
0x00 00 01 a0: 82 e1 30 15 b7 0c ac 90  e3 44 7b 6b ef e4 00 63
0x00 00 01 b0: 61 9f 2a c4 b6 35 77 a6  aa 92 0c c7 e1 55 a5 eb
0x00 00 01 c0: 22 18 8b 5b d9 6a 0c ad  de f4 db bb b4 43 ee 5d
0x00 00 01 d0: e2 cb 16 8e 32 ec f3 15  72 9a e7 bd be 0b 9d 42
```

   Welchen Inhalt hat der Cache nach Ausführung des Programms? Nummerieren Sie die Mengen entsprechend deren Index!

   Es genügt, jeweils das erste und letzte Byte der Daten im Cache-Block einzutragen. 6 Punkte

Menge	Block	Valid	Tag	Daten
	0			
	1			
	2			
	3			
	4			
	5			
	6			
	7			

Zusätzlicher Platz